HPC-AI-Report2

December 4, 2024

1 UM6P hackathon report (part 2)

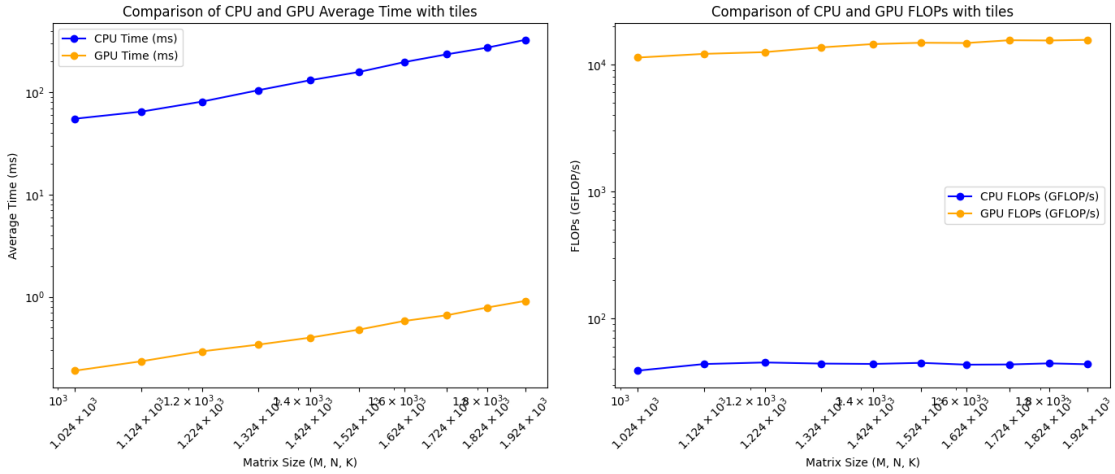
1.1 TajTech

1.1.1 Mundiapolis

[4]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

[5]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the new CSV data
file_path = 'gemmtotask3.csv'
data = pd.read_csv(file_path)

# Extract the relevant columns
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']
cpu_flops = data['PerformanceCPU(GFLOP/s)']
gpu_flops = data['PerformanceGPU(GFLOP/s)']

# Create the figure and subplots
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU vs GPU average times on the first subplot with log scale
ax[0].plot(matrix_sizes, cpu_times, label='CPU Time (ms)', color='blue',␣
 ↪marker='o')
ax[0].plot(matrix_sizes, gpu_times, label='GPU Time (ms)', color='orange',␣
 ↪marker='o')
ax[0].set_title('Comparison of CPU and GPU Average Time with tiles',␣
 ↪fontsize=12)
ax[0].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[0].set_ylabel('Average Time (ms)', fontsize=10)
ax[0].set_xscale('log')
```

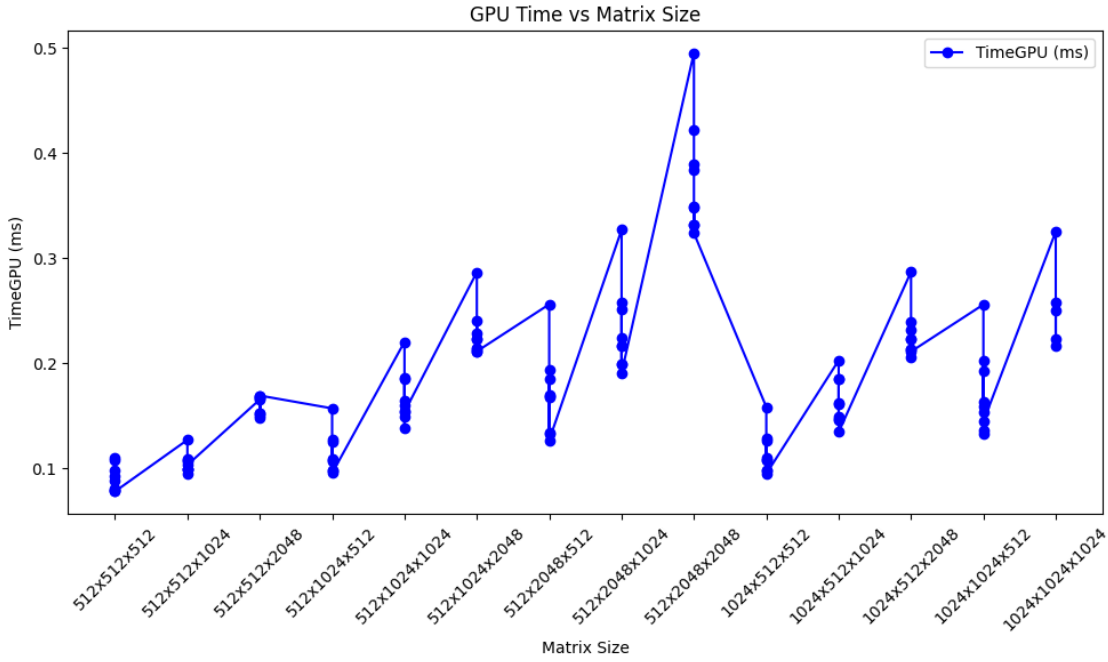
```
ax[0].set_yscale('log')
ax[0].set_xticks(matrix_sizes)  # Ensure proper scaling on x-axis
ax[0].tick_params(axis='x', rotation=45)  # Rotate x-axis labels by 45 degrees
ax[0].legend()

# Plot CPU vs GPU FLOPs on the second subplot with log scale
ax[1].plot(matrix_sizes, cpu_flops, label='CPU FLOPs (GFLOP/s)', color='blue',␣
 ↪marker='o')
ax[1].plot(matrix_sizes, gpu_flops, label='GPU FLOPs (GFLOP/s)',␣
 ↪color='orange', marker='o')
ax[1].set_title('Comparison of CPU and GPU FLOPs with tiles', fontsize=12)
ax[1].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[1].set_ylabel('FLOPs (GFLOP/s)', fontsize=10)
ax[1].set_xscale('log')
ax[1].set_yscale('log')
ax[1].set_xticks(matrix_sizes)  # Ensure proper scaling on x-axis
ax[1].tick_params(axis='x', rotation=45)  # Rotate x-axis labels by 45 degrees
ax[1].legend()

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```

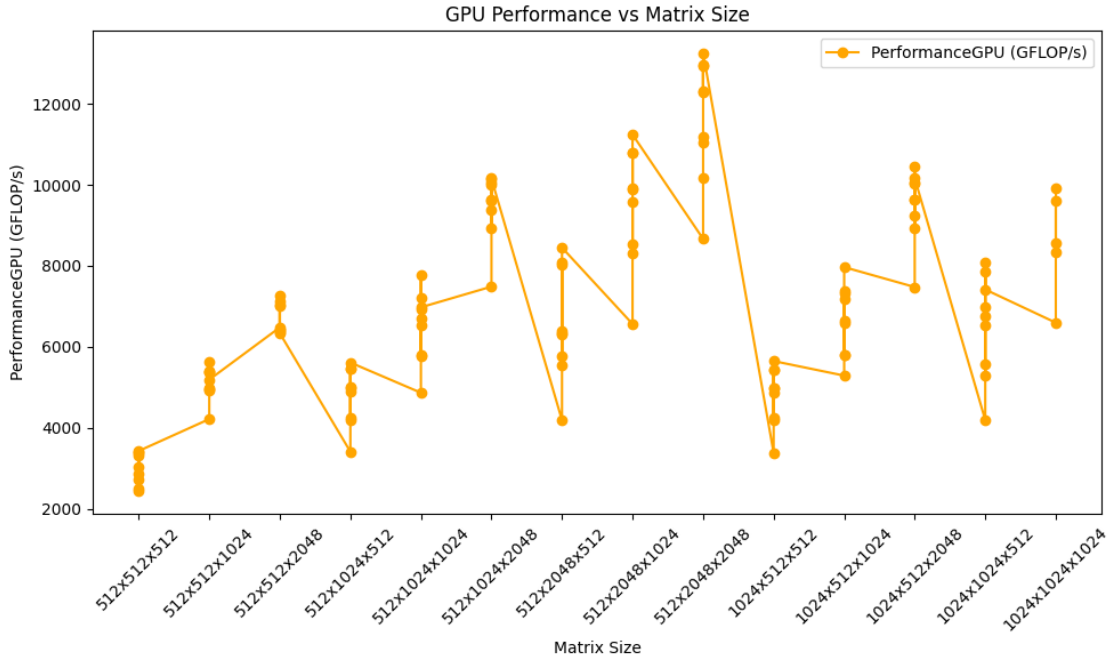


[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame
df = pd.read_csv('gemmTileBatchTests.csv')
```

```
# Create a new column 'Matrix Size' by combining M, N, K
df['Matrix Size'] = df['M'].astype(str) + 'x' + df['N'].astype(str) + 'x' +␣
 ↪df['K'].astype(str)

# Plot 1: TimeGPU vs Matrix Size
plt.figure(figsize=(10, 6))
plt.plot(df['Matrix Size'], df['TimeGPU(ms)'], marker='o', label='TimeGPU␣
 ↪(ms)', color='blue')
plt.title('GPU Time vs Matrix Size')
plt.xlabel('Matrix Size')
plt.ylabel('TimeGPU (ms)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Time_vs_Matrix_Size.png')
plt.show()

# Plot 2: PerformanceGPU vs Matrix Size
plt.figure(figsize=(10, 6))
plt.plot(df['Matrix Size'], df['PerformanceGPU(GFLOP/s)'], marker='o',␣
 ↪label='PerformanceGPU (GFLOP/s)', color='orange')
plt.title('GPU Performance vs Matrix Size')
plt.xlabel('Matrix Size')
plt.ylabel('PerformanceGPU (GFLOP/s)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Performance_vs_Matrix_Size.png')
plt.show()
```

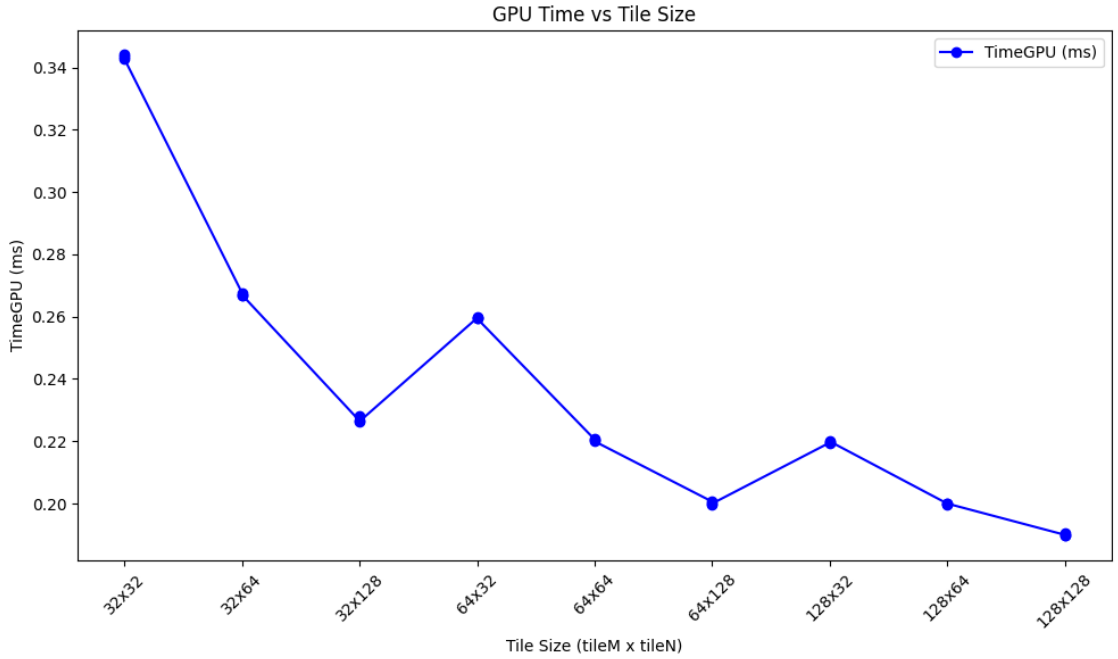
```
/home/abdennacer/.local/lib/python3.10/site-
packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import
Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g.
as a system package and as a pip package). As a result, the 3D projection is not
available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions
of "
```

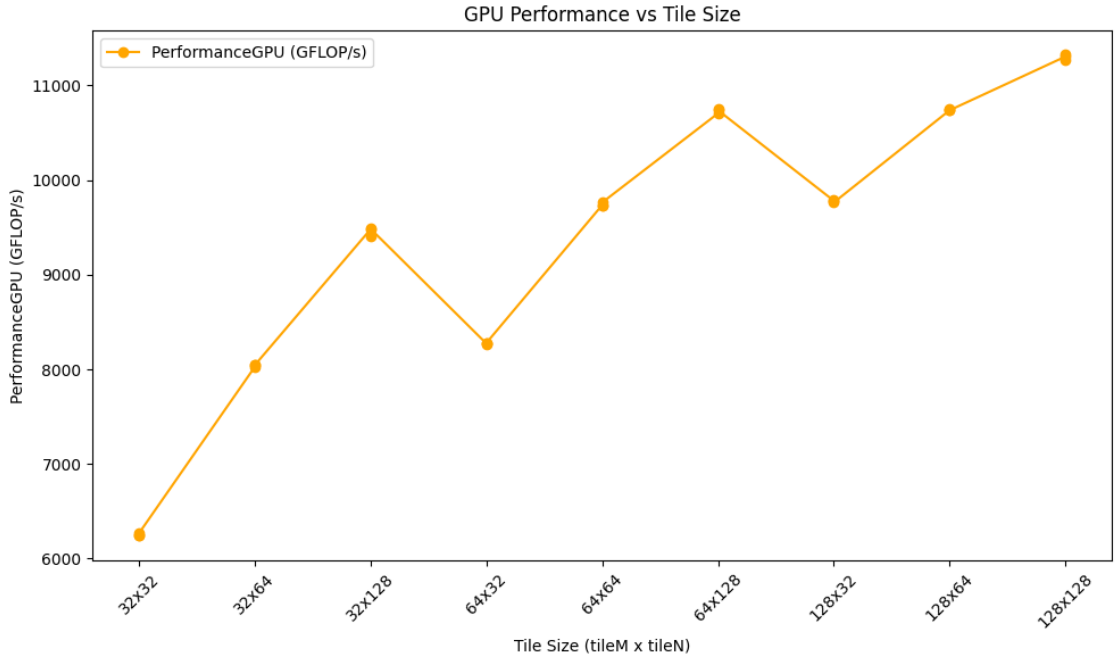




[2]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the fixed matrix size CSV file into a DataFrame
```

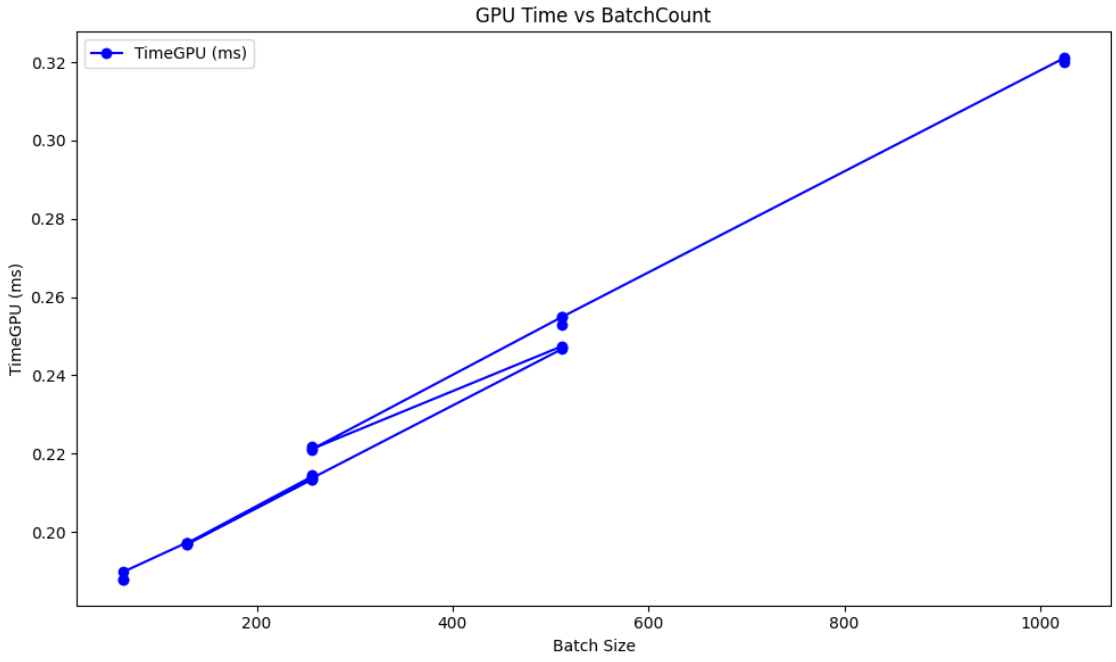
```
df = pd.read_csv('gemmTileBatchTestsFixedMNK.csv')

# Create a new column 'Tile Size' by combining tileM and tileN
df['Tile Size'] = df['tileM'].astype(str) + 'x' + df['tileN'].astype(str)

# Plot 1: TimeGPU vs Tile Size
plt.figure(figsize=(10, 6))
plt.plot(df['Tile Size'], df['TimeGPU(ms)'], marker='o', label='TimeGPU (ms)',␣
 ↪color='blue')
plt.title('GPU Time vs Tile Size')
plt.xlabel('Tile Size (tileM x tileN)')
plt.ylabel('TimeGPU (ms)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Time_vs_Tile_Size.png')
plt.show()

# Plot 2: PerformanceGPU vs Tile Size
plt.figure(figsize=(10, 6))
plt.plot(df['Tile Size'], df['PerformanceGPU(GFLOP/s)'], marker='o',␣
 ↪label='PerformanceGPU (GFLOP/s)', color='orange')
plt.title('GPU Performance vs Tile Size')
plt.xlabel('Tile Size (tileM x tileN)')
plt.ylabel('PerformanceGPU (GFLOP/s)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Performance_vs_Tile_Size.png')
plt.show()
```

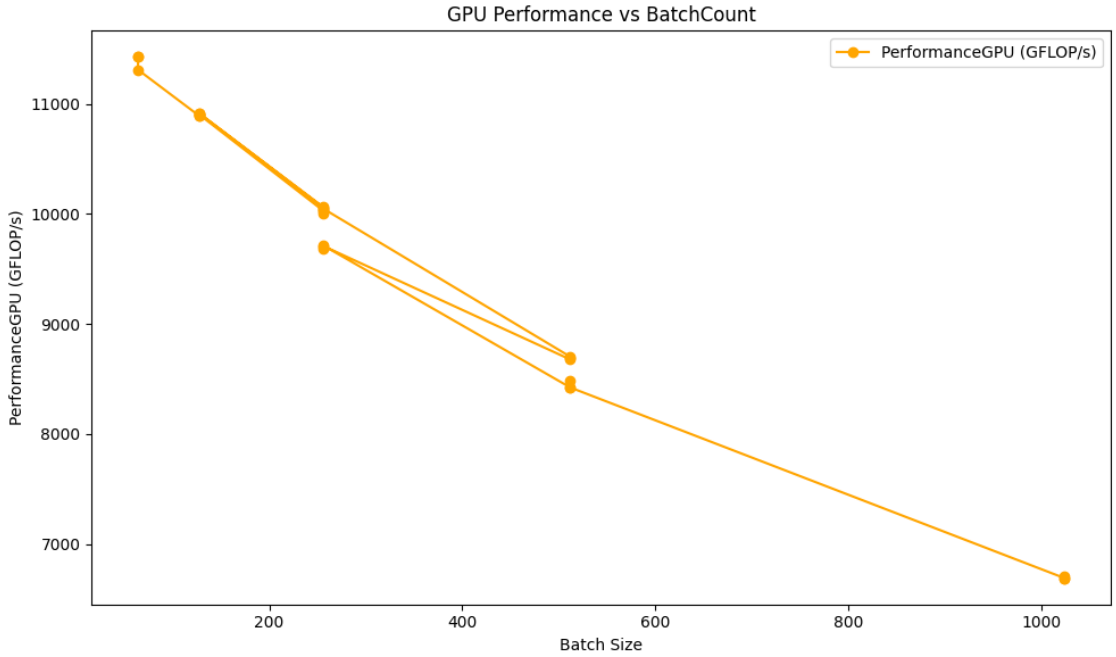




[3]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame
```

```
df = pd.read_csv('gemmTileBatch_PerformanceByBatchcount.csv')

# Plot 1: TimeGPU vs BatchCount
plt.figure(figsize=(10, 6))
plt.plot(df['BatchCount'], df['TimeGPU(ms)'], marker='o', label='TimeGPU (ms)',␣
 ↪color='blue')
plt.title('GPU Time vs BatchCount')
plt.xlabel('Batch Size')
plt.ylabel('TimeGPU (ms)')
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Time_vs_BatchCount.png')
plt.show()

# Plot 2: PerformanceGPU vs BatchCount
plt.figure(figsize=(10, 6))
plt.plot(df['BatchCount'], df['PerformanceGPU(GFLOP/s)'], marker='o',␣
 ↪label='PerformanceGPU (GFLOP/s)', color='orange')
plt.title('GPU Performance vs BatchCount')
plt.xlabel('Batch Size')
plt.ylabel('PerformanceGPU (GFLOP/s)')
plt.tight_layout()
plt.legend()
plt.savefig('GPU_Performance_vs_BatchCount.png')
plt.show()
```




GPU Performance vs BatchCount
PerformanceGPU (GFLOP/s)
11000
10000
9000
8000
7000
PerformanceGPU (GFLOP/s)
200
400
600
800
1000
Batch Size

[ ]: